

(19) 日本国特許庁(JP)

(12) 公表特許公報(A)

(11) 特許出願公表番号

特表2004-531878

(P2004-531878A)

(43) 公表日 平成16年10月14日(2004.10.14)

(51) Int.Cl.⁷
H01L 21/02F 1
H01L 21/02

テーマコード(参考)

Z

審査請求 未請求 予備審査請求 有 (全 42 頁)

(21) 出願番号 特願2002-568120 (P2002-568120)
 (86) (22) 出願日 平成13年10月22日 (2001.10.22)
 (85) 開示文提出日 平成15年8月21日 (2003.8.21)
 (86) 國際出願番号 PCT/US2001/050178
 (87) 國際公開番号 WO2002/069063
 (87) 國際公開日 平成14年9月6日 (2002.9.6)
 (31) 優先権主張番号 09/789,871
 (32) 優先日 平成13年2月21日 (2001.2.21)
 (33) 優先権主張国 米国(US)

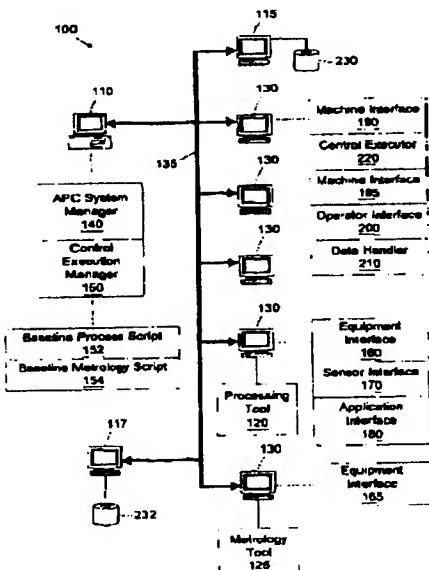
(71) 出願人 591016172
 アドバンスト・マイクロ・ディバイシズ・
 インコーポレイテッド
 ADVANCED MICRO DEVI
 CES INCORPORATED
 アメリカ合衆国、94088-3453
 カリフォルニア州、サンディベール、ビイ・
 オウ・ボックス・3453、ワン・エイ・
 エム・ディ・ブレイス、メイル・ストップ
 ・68 (郵便なし)
 (74) 代理人 100099324
 弁理士 鈴木 正剛
 (74) 代理人 100111615
 弁理士 佐野 良太

最終頁に続く

(54) 【発明の名称】ベースライン制御スクリプトを用いてツールを制御するための方法および装置

(57) 【要約】

製造システム(100)を制御するための方法が、複数のツール(120, 125)で被加工物を処理し、この複数のツール(120, 125)のうちの選択したツール(120, 125)についてベースライン制御スクリプト(152, 154)を開始し、ベースライン制御スクリプト(152, 154)のコンテキスト情報を提供し、コンテキスト情報に基づいてツールのタイプを判定し、選択したツール(120, 125)の制御ルーチンをツールのタイプに基づいて選択し、制御ルーチンを実行して選択したツール(120, 125)のコントロールアクションを生成することを含む。製造システム(100)は、被加工物を処理するように作られた複数のツール(120, 125)と、制御実行マネージャ(150)と、コントロールエグゼキュータ(220)とを含む。制御実行マネージャ(150)は、複数のツール(120, 125)のうちの選択したツール(120, 125)についてベースライン制御スクリプト(152, 154)を開始し、ベースライン制御スクリプト(152, 154)のコンテキスト情報を提供するように作ら



【特許請求の範囲】**【請求項 1】**

複数のツール（120、125）で被加工物を処理する際に、
前記複数のツール（120、125）のうちの選択したツール（120、125）についてベースライン制御スクリプト（152、154）を開始する処理と、
前記ベースライン制御スクリプト（152、154）のコンテキスト情報を提供する処理
と、
前記コンテキスト情報に基づいてツールのタイプを判定する処理と、
前記選択したツール（120、125）の制御ルーチンをツールのタイプに基づいて選択
する処理と、
前記制御ルーチンを実行して前記選択したツール（120、125）のコントロールアク
ションを生成する処理とを含む、製造システム（100）を制御するための方法。
10

【請求項 2】

前記制御ルーチンを選択する処理が、制御ルーチンのライブラリ（240）にリンクする
処理をさらに含む、請求項1に記載の方法。

【請求項 3】

前記コンテキスト情報が前記選択したツールに関連したエンティティ識別コードを含み、
前記ツールのタイプを判定する処理が前記エンティティ識別コードに基づいてツールのタ
イプを判定する処理をさらに含む、請求項1に記載の方法。
20

【請求項 4】

前記コンテキスト情報がオペレーション識別コードを含み、前記制御ルーチンを選択する
処理が前記ツールのタイプと前記オペレーション識別コードとにに基づいて制御ルーチンを
選択する処理をさらに含む、請求項1または3に記載の方法。

【請求項 5】

前記コンテキスト情報が製品識別コードを含み、前記制御ルーチンを選択する処理がツー
ルのタイプと前記製品識別コードとにに基づいて制御ルーチンを選択する処理をさらに含む
、請求項1、3または4に記載の方法。

【請求項 6】

被加工物を処理するように作られた複数のツール（120、125）と、
前記複数のツール（120、125）のうちの選択したツール（120、125）について 30
ベースライン制御スクリプト（152、154）を開始し、前記ベースライン制御スクリ
プト（152、154）のコンテキスト情報を提供するように作られた制御実行マネー
ジャー（150）と、
前記ベースライン制御スクリプト（152、154）を実行し、前記コンテキスト情報に
に基づいてツールのタイプを判定し、前記選択したツール（120、125）の制御ルーチ
ンをツールのタイプに基づいて選択し、前記制御ルーチンを実行して前記選択したツール
（120、125）のコントロールアクションを生成するように作られたコントロールエ
グゼキュータ（220）とを含む、製造システム（100）。

【請求項 7】

前記コントロールエグゼキュータ（220）が制御ルーチンのライブラリ（240）にリ 40
ンクして制御ルーチンを選択するように作られている、請求項6に記載のシステム（10
0）。

【請求項 8】

前記コンテキスト情報が前記選択したツールに関連したエンティティ識別コードを含み、
コントロールエグゼキュータ（220）が前記エンティティ識別コードに基づいてツール
のタイプを判定するように作られている、請求項6に記載のシステム（100）。

【請求項 9】

前記コンテキスト情報がオペレーション識別コードを含み、前記コントロールエグゼキュ
ータ（220）がツールのタイプと前記オペレーション識別コードとにに基づいて制御ル
ーチンを選択するように作られている、請求項6または8に記載のシステム（100）。
50

【請求項 10】

前記コンテキスト情報が製品識別コードを含み、前記コントロールエグゼキュータ（220）がツールのタイプと前記製品識別コードとに基づいて制御ルーチンを選択するようになされている、請求項6、8または9に記載のシステム（100）。

【発明の詳細な説明】**【技術分野】****【0001】**

本発明は広義には半導体デバイス製造の分野に関し、特に、ベースライン制御スクリプトを用いてツールを制御するための方法および装置に関する。

10

【背景技術】**【0002】**

半導体業界では、マイクロプロセッサ、メモリデバイスなどの集積回路デバイスの品質、信頼性、スループットを向上させたいという強い要求が常にある。この要求は、それまでよりも確実に動作する高品質のコンピュータや電子デバイスを求める消費者需要を受けてなお一層高まっている。こうした需要があることから、トランジスタなどの半導体デバイスの製造ならびにこのようなトランジスタを組み入れた集積回路デバイスの製造の点でたゆみない改善がなされてきている。また、典型的なトランジスタの構成部品の製造時に欠陥が少なくなることで、トランジスタ1つあたりの総コストの削減になるほか、このようなトランジスタを組み入れた集積回路デバイスのコストの削減にもなる。

20

【0003】

通常、フォトリソグラフィ用ステッパー、エッチツール、成膜ツール、研磨ツール、高速熱処理ツール、インプランテーションツールなどを含むさまざまな処理ツールを利用して、一連の処理ステップが多数のウエハ上で実施される。ここ数年、半導体処理ツールの基礎をなすいくつかの技術が次第に注目されるようになり、相当な改良がなされている。しかしながら、この分野でこうした進歩があつたにもかかわらず、現段階で商用入手可能な処理ツールの多くにはある種の欠点がある。特に、このようなツールには、ユーザが利用しやすい形式でパラメータデータの履歴を提供する機能や、イベントログの記録、最新の処理パラメータとラン（実行）全体の処理パラメータのリアルタイムでのグラフィック表示、遠隔地すなわちローカルサイトおよび世界中で監視を行う機能などの高度なプロセスデータ監視能力が欠けていることが多い。これらの欠点が原因で、スループットや精度、安定性、再現性などの重要な処理パラメータ、処理温度、機械的なツールのパラメータなどが最適とはいえない形で制御される結果になりかねない。この多様性はラン内での格差、ランとランとの格差、ツールとツールとの格差として表れ、こうした格差が進むと製品の品質や性能の偏りへつながる可能性があるのに対し、これらのツールのための理想的な監視・診断システムがあれば重要なパラメータの制御を最適化するための手段が得られるだけでなく上記の多様性を監視する手段も得られるであろう。

30

【0004】

半導体処理ラインのオペレーションを改善するためのひとつの手法として全工場規模の制御システムを採用してさまざまな処理ツールのオペレーションを自動制御することがあげられる。製造ツールが製造フレームワークと通信または処理モジュールのネットワークと通信する。それぞれの製造ツールは一般に機器インターフェースと接続されている。機器インターフェースは、製造ツールと製造フレームワークとの間の通信をしやすくするマシンインターフェースと接続されている。マシンインターフェースは通常、高度プロセス制御（APC）システムの一部であってもよいものである。APCシステムは製造モデルに基づいて制御スクリプトを開始するが、この制御スクリプトは製造プロセスを実行するのに必要なデータを自動的に取り込むソフトウェアプログラムであってもよい。しばしば半導体デバイスは複数の製造ツールによって複数のプロセスで段階を経て作られ、処理後の半導体デバイスの品質に關係するデータが生成される。

40

【0005】

製作プロセスの間、製作対象となるデバイスの性能に影響するさまざまな出来事が起こる

50

可能性がある。つまり、製作プロセスステップのばらつきが原因でデバイス性能のばらつきが発生する。構造の臨界寸法、ドーピングレベル、接触抵抗、粒子汚染などの要因がすべてデバイスの最終性能に影響する可能性を持つことがある。処理ラインのツールはそれぞれ処理のばらつきを減らすべく性能モデルに従って制御されている。普通に制御されるツールとしては、フォトリソグラフィ用ステッパー、研磨ツール、エッチングツール、成膜ツールがあげられる。これらのツール用のプロセスコントローラには前処理および/または後処理測定データが供給される。処理時間などの動作レシピパラメータについては、性能モデルと計測情報をもとにプロセスコントローラで計算し、標的値にできるだけ近い処理結果を達成しようと試みられている。このようにしてばらつきを減らすことで、スループットを向上させ、コストを削減し、デバイス性能をそれまでよりも高くできるなどの結果につながるが、これらはいずれも収益性の向上に結び付く。10

【発明の開示】

【発明が解決しようとする課題】

【0006】

コンフィギュレーション制御と効率の問題は、全工場規模のAPCシステムなどの分散コンピューティング環境では普通に見られるものである。一般に、プロセスコントローラを構築するための制御コードは多くのソフトウェア開発者によって書かれている。個々の開発者が特定タイプのコントローラを開発しつつ広範囲にわたって作業をすることもある。彼ら開発者がそれぞれ独自のプログラミングスタイルを持ち、自分で作り出したルーチンに頼っていることは珍しくない。たとえば、APCフレームワーク内のデータベースや他のエンティティと接続し、さまざまな数学関数や基本のユーティリティ関数を実行するためのルーチンのセットを開発者ごとに持っている場合がある。20

【0007】

このような形態をとることに伴うひとつの問題として、プロセス制御スクリプト間の一貫性がほとんどない点があげられる。開発者独自のスクリプトが多数あることもコンフィギュレーション制御の問題や効率面での問題の原因となる。開発者らは、すでに別の開発者が違うタイプのプロセスコントローラ向けに開発したであろうものと同様のコードを書くのに相当な時間を費やす場合がある。標準から外れたコードのデバッグにもさらに多くの時間を要し、余計に効率が悪くなる。30

【0008】

本発明は、上述した問題のうちひとつまたはそれ以上の影響を解決するか少なくとも低減しようとするものである。

【課題を解決するための手段】

【0009】

発明の開示

本発明の一態様は、製造システムを制御するための方法に見られる。この方法は、複数のツールで被加工物を処理し、この複数のツールのうちの選択したツールについてベースライン制御スクリプトを開始し、ベースライン制御スクリプトのコンテキスト情報を提供し、このコンテキスト情報に基づいてツールのタイプを判定し、選択したツールの制御ルーチンをツールのタイプに基づいて選択し、制御ルーチンを実行して選択したツールのコントロールアクションを生成することを含む。40

【0010】

本発明のもうひとつの態様は、被加工物を処理するように作られた複数のツールと、制御実行マネージャと、コントロールエグゼキュータとを含む製造システムに見られる。制御実行マネージャは、複数のツールのうちの選択したツールについてベースライン制御スクリプトを開始し、ベースライン制御スクリプトのコンテキスト情報を提供するように作られている。コントロールエグゼキュータは、ベースライン制御スクリプトを実行し、このコンテキスト情報に基づいてツールのタイプを判定し、選択したツールの制御ルーチンをツールのタイプに基づいて選択し、制御ルーチンを実行して選択したツールのコントロールアクションを生成するように作られている。50

【0011】

添付の図面を参照して以下の説明を参照することで本発明を理解することができよう。図中、同様の構成要素には同様の参照符号を付してある。

【0012】

本発明はさまざまな形に改変および変更が可能なものであるが、一例としてその特定の実施形態を図示し、本願明細書中にて詳細に説明してある。しかしながら、特定の実施形態に関する本願明細書の記載がここに開示の特定の形態に本発明を限定することを意図したものではなく、添付の特許請求の範囲に規定の本発明の範囲に含まれる改変例、等価物および代わりとなる物をすべて包含することを意図している点を理解されたい。

10

【発明を実施するための最良の形態】**【0013】****発明の実施の形態**

以下、本発明の実施形態について説明する。明瞭な説明とするために、本願明細書では実際に使われる実装の構造すべてについて説明してあるとは限らない。もちろん、このような実際の実施形態を開発するにあたって、実装ごとに異なるシステム関連の制約およびビジネス関連の制約に従うなど、実装ごとに固有の多数の決定を行い、開発者ごとに異なる目標を達成しなければならない点は理解できよう。さらに、このような開発の取り組みが複雑で時間を要するものとなる可能性があるとはいえ、本願開示の利益を享受する当業者の日常作業になろうものであることも理解できよう。

20

【0014】

以下、図面を参照するが、まず図1を参照すると、高度プロセス制御（A P C）システム100の簡略ブロック図が示されている。A P Cシステム100は、ランとランとの間の制御や故障の検出／分類を可能にしている取り替え可能な標準化ソフトウェアコンポーネントからなる分散ソフトウェアシステムである。これらのソフトウェアコンポーネントは、半導体製造装置・材料に関する業界団体（S E M I）コンピュータ統合生産（C I M）フレームワーク対応のシステム技術仕様および高度プロセス制御（A P C）フレームワークに基づくアーキテクチャ標準を満たしている。C I M（C I Mフレームワークドメインアーキテクチャに関するS E M I E 8 1 - 0 6 9 9 - 暫定仕様書）およびA P C（C I Mフレームワーク高度プロセス制御コンポーネントに関するS E M I E 9 3 - 0 9 9 9 - 暫定仕様書）の各仕様についてはS E M Iから一般に入手可能である。この特有のアーキテクチャは、オブジェクト指向プログラミングを利用しているソフトウェアに極めて大きく依存し、分散オブジェクトシステム向けにオブジェクトマネジメントグループ（O M G）が策定した共通オブジェクトリクエストプロトコルアーキテクチャ（C O R B A）およびC O R B Aサービスの仕様を採用している。このO M G C O R B Aアーキテクチャについての情報および仕様は容易に一般に入手可能である。本願明細書にて説明するようなA P Cシステム100の機能を果たすようにすることのできるソフトウェアシステムの一例として、ケールエー・テンコール・インコーポレイテッド（K L A - T e n c o r , I n c . ）から提供されているC a t a l y s tシステムがあげられる。

30

【0015】

これらのコンポーネントはC O R B Aインタフェース定義言語（I D L）を使って互いに通信し、その対話をサポートするための共通のサービスセットに頼っている。分散オブジェクトサービスの標準セットはO M Gによって規定されている。これらのサービスには以下のものがある。

40

【0016】

C O R B A - コンポーネントとコンポーネントとの間での直接対話に必ず使われる標準ベースの通信プロトコルである。オブジェクト指向の遠隔呼出し通信モデルに従って標準インターフェースを定義することができる。これらのインターフェースとすべてのA P C通信はI D Lを使って定義される。コンポーネントは互いに他のインターフェースでオペレーションを呼び出すことによって通信する。コンポーネント間ではオペレーションパラメータおよび返り値としてデータの受け渡しがなされる。

50

【0017】

OMGイベントサービスーコンポーネント間での非同期的な通信をサポートするものである。APCオブジェクトの多くは状態が変化したときにイベントを発信する。これらのイベントは、該当するイベントサブスクリーブによって受信される。APCシステム内でのイベント利用の例としては、通信コンポーネントの状態（エラーハンドルを含む）、故障の検出および分類ソフトウェアで検出した故障アラームの通知、マシンのステータスと収集したデータの報告があげられるが、これに限定されるものではない。

【0018】

OMGトレーディングサービスーコンポーネントが自己と対話すべき別のコンポーネントを見つけることができるようになるものである。コンポーネントをインストールすると、そのサービスの説明（サービスのオファー）がトレーディングサービスにエクスポートされる。別のコンポーネントが特定の規準を満たすサービスプロバイダの一覧を後から要求することができる。トレーディングサービスでは、要求されたサービスを提供できる他のコンポーネントの一覧を提供する。この機能は、コンポーネントの起動時にひとつのコンポーネントが自己と対話しなければならない他のコンポーネントを見つけることができるようになる目的で使われる。また、これはプラン実行コンポーネントがプランに指定された必要な能力を提供するのにケイバビリティプロバイダ（Capability Provider）を見つけなければならないプランスタートアップ（Plan Startup）にも使われる。

10

【0019】

これらのサービスは従来技術において周知である。OMGのCORBA/IOP仕様の文書およびCORBAサービス仕様の文書は当業者に広く配布されており、そこにはさらに詳細な説明がなされている。

20

【0020】

図示の実施形態では、APCシステム100は半導体製造環境を制御できるように作られている。複数のコンポーネントがCORBAインターフェース定義言語（IDL）を使って互いに通信する。連携しているソフトウェアコンポーネントは、プロセス制御プラン／ストラテジーを管理し、プロセス機器、計測ツール、アドオンセンサからデータを収集し、この情報をを利用してさまざまなプロセス制御アプリケーション／アルゴリズムを呼び出し、プロセスモデルを更新し、該当する場合はツール動作レシピバラメータを修正／ダウングレードする。APCシステム100は半導体生産プロセスを制御するための全工場規模のソフトウェアシステムであるが、これは本発明を実施するにあたって必要なことではない。本発明が教示するストラテジーは、任意の規模で異なるコンピュータシステムに適用可能なものである。

30

【0021】

代表的な実装例では、APCシステム100は、APCホストコンピュータ110と、データベースサーバ115、117と、処理ツール120と、計測ツール125と、1つまたはそれ以上のワークステーション130とを含む。APCシステムの各コンポーネントはバス135を介して相互に接続されている。バス135は実際は複数の層で構成されて複数のプロトコルを利用するものであってもよい。APCシステム100の全体としてのオペレーションはAPCホストコンピュータ110上に常駐するAPCシステムマネージャ140によって指示されている。APCシステムマネージャ140は、APCフレームワーク用に開発されたすべてのサーバの管理サービス、コンフィギュレーションサービス、イベントサービス、状態サービス；APCシステム100内のコンポーネントの定義、グルーピング、インストール、マネジメント；診断および監視の目的でアクティビティおよびトレース情報を捕捉するための中央サービス；セットアップ値、システム環境設定を含むコンポーネントコンフィギュレーション情報の中央の収納場所；従属オブジェクトおよびイベントチャネルのリストを提供するものである。しかしながら、別の実施形態では、これらの機能を、ベースマネージャ、システムマネージャ、ロガー、レジストリなどの1つまたはそれ以上のソフトウェアコンポーネントに分割してもよい。

40

50

【0022】

APCシステム100は複数の処理モジュールからなるネットワークを含む。これらの処理モジュールは「インテグレーションコンポーネント」と呼ばれることがある。インテグレーションコンポーネントは、既存のファクトリーシステムとのインターフェースとして機能し、APCプランを走らせることができるようにするものである。「APCプラン」とは、詳細については後述するように特定のタスクを実行するために呼ばれるアプリケーションプログラムのことである。インテグレーションコンポーネントについては、APCシステム100内のさまざまな処理資源によって受け入れられる場合があるものとして示してある。これらの特定のホスティングロケーションは例示の目的で提供されている。処理リソース同士は相互に接続されており、システムの複雑さに応じてさまざまなソフトウェアコンポーネントをさまざまなコンピュータに分散させてもよいし、中央に集中させてもよい。この特定の実施形態におけるインテグレーションコンポーネントは各々ソフトウェア的に実装されている。これらのコンポーネントは従来技術において周知のオブジェクト指向プログラミングの手法を使ってC++でプログラムされている。APCシステム100の利点のひとつにモジュール構造であるという点があるが、これによってソフトウェアコンポーネントの移植性が得られる。インテグレーションコンポーネントは、APCシステムマネージャ140、制御実行マネージャ150、ツール120、125に関連した機器インターフェース160、165、処理ツール120に関連したセンサインターフェース170、アプリケーションインターフェース180、マシンインターフェース190、195、オペレータインターフェース200、データハンドラ210を含むがこれに限定されるものではない。
10 20

【0023】

制御実行マネージャ150は、APCシステム100のオペレーションの「振り付け」を主に担うコンポーネントである。制御実行マネージャ150はAPCプランを解釈し、メインスクリプトおよびサブスクリプトを実行し、イベントの命令時にイベントスクリプトを呼び出す。多様なプラン、スクリプト、サブスクリプトをさまざまな実装に用いることができる。さまざまなプラン、スクリプト、サブスクリプトの具体的な数と機能はそれぞれの実装内容に依存する。たとえば、本実施形態は、
データ収集プラン—特定の処理機器からどのデータを収集すべきか、そのデータをどのようにして折り返し報告するかという要件を定義するセンサおよびマシンインターフェースで用いられるデータ構造
期間プラン—データ収集の開始、データ収集の終了など、センサを作動させるトリガー条件とトリガー遅延とを定義するプラン
レポートプラン—収集したデータをどのようにすべきかとデータの可用性をいつ信号伝達すべきかを定義するプラン
サンプリングプラン—外部センサによってデータを収集する頻度を定義するプラン
制御プラン—APCアクティビティを実行するために一緒に使われるよう設計された制御スクリプトの集合
制御スクリプト—定義された特定の状況下でAPCシステムが実行する一連のアクション／アクティビティ
などのプランを含むがこれに限定されるものではない。
30 40

【0024】

制御実行マネージャ150は、処理ツール120などの特定のツールについて、すべてのインテグレーションコンポーネント内でユーザが定義したプロセス制御プランの実行をコーディネートする。指示があると、制御実行マネージャ150はプランとこれに関連したスクリプトとを読み出す。次に、サブスクリプトを前処理してメインスクリプトおよびイベントスクリプトへのルーチンを提供する。また、プランに指定されている、プランを実行するのに必要な能力の一覧を取得し、必要な能力が得られるしかるべきインテグレーションコンポーネントに接続する。

【0025】

次に、制御実行マネージャ150は、このプランを走らせる責任をコントロールエグゼキュー¹⁰タ220に委ねる。図示の実施形態では、制御実行マネージャ150はベースライン制御スクリプトを使って実施すべきコントロールアクションを判定する。ベースラインプロセススクリプト152は処理ツール120などの処理ツールと併用すべきものであり、ベースライン計測スクリプト154は計測ツール125などの計測ツールと併用すべきものである。ベースラインスクリプト152、154の詳細については、図2乃至図6を参照して後述する。

【0026】

制御実行マネージャ150は、プランを順次実行してプランの終了またはプラン実行時に²⁰おけるエラーを制御実行マネージャ150に折り返し報告するためのコントロールエグゼキュー²⁰タ220を、適当なベースラインプロセススクリプト152またはベースライン計測スクリプト154に基づいて作成する。このように、制御実行マネージャ150が実行されるすべてのプランの全体としてのマネジメントを担っているのに対し、各コントロールエグゼキュー²²⁰タ220はひとつのプランを走らせることだけを担っている。コントロールエグゼキュー²²⁰タ220は制御実行マネージャ150によって作成され、プランが生きている間は存在し、そのプランが終了または異常終了した旨の報告の後で制御実行マネージャ150によって破棄される。制御実行マネージャ150は、複数のコントロールエグゼキュー²²⁰タ220を介して複数のプランを同時に始めることができる。

【0027】

マシンインタフェース190、195は、APCシステムマネージャ140などのAPC²⁰フレームワークと機器インタフェース160、165との間のギャップを埋めるものである。マシンインタフェース190、195は、処理ツールまたは計測ツール120、125とAPCフレームワークとを接続し、マシンのセットアップ、起動、監視、データ収集をサポートする。この特定の実施形態では、マシンインタフェース190、195は主に機器インタフェース160、165の特定の通信とAPCフレームワークのCORBA通信との間の翻訳を行う。特に、マシンインタフェース190、195は、コマンド、ステータスイベント、収集されたデータを機器インタフェース160、165から受け取り、必要に応じてこれを他のAPCコンポーネントおよびイベントチャネルに転送する。今度は、他のAPCコンポーネントからの応答がマシンインタフェース190、195で受信され、機器インタフェース160、165に送られる。また、マシンインタフェース190、195は、必要に応じてメッセージとデータを再フォーマットおよび再構築する。³⁰マシンインタフェース190、195は、APCシステムマネージャ140内のスタートアップ/シャットダウン手順をサポートする。また、機器インタフェース160、165によって収集されたデータを一時的に蓄積し、適切なデータ収集イベントを発信するAPCデータコレクタとしても機能する。

【0028】

センサインタフェース170は、処理ツール120のオペレーションを監視しているセンサが生成したデータを収集する。センサインタフェース170は、ラボビュー(LabVIEW)または他のセンサなどの外部センサやバスをベースにしたデータ取得ソフトウェアと通信するための適当なインタフェース環境を提供するものである。アプリケーションインタフェース180は、ラボビュー(LabVIEW)、マセマティカ(Mathematica)、モデルウェア(ModelWare)、マトラボ(MatLab)、シムカ(Simca)4000、エクセル(Excel)などの制御プラグインアプリケーションを実行するための適切なインタフェース環境を提供するものである。センサについて⁴⁰は相手先商標製品製造(OEM)によって処理ツール120と一緒に供給してもよいし、OEMから入手した後でインストールされる「アドオン」センサであってもよい。センサインタフェース170はセンサによって生成されたデータを収集する。このセンサは、たとえば動作条件の圧力および温度についてのデータを生成するものであっても構わない。アプリケーションインタフェース180はコントロールエグゼキュー²²⁰タ220からデータを得て、そのデータに関する計算または分析を実行する。次に、結果をコントロールエグ⁵⁰

ゼキュータ220に返す。マシンインタフェース190およびセンサインタフェース170は、使われるデータを収集するための共通の機能一式を利用する。機器インタフェース160は処理ツール120についてセンサが収集したそれぞれのデータを集め、集めたデータをマシンインタフェース190に送信する。

【0029】

オペレータインタフェース200は、グラフィカルユザインターフェース(GUI) (図示せず) によってウエハ製作技術者とAPCシステム100との通信を容易にするものである。このGUIはWindows(R) またはUNIXベースのオペレーティングシステムであっても構わないが、これは本発明を実施するにあたって必要なことではない。特に、いくつかの別の実施形態ではGUIを採用することすらせず、ディスクオペレーティングシステム(DOS)ベースのオペレーティングシステムを使って通信を行うことができる。オペレータインタフェース200はダイアログボックスを表示して情報を提供し、指示を求め、別のデータを収集する。CORBAインタフェースによって、技術者らはオペレータインタフェース200のコンポーネントを使って任意の数のディスプレイ群に多様なポップアップダイアログを同時に表示することができる。また、オペレータインタフェース200は、ポップアップを表示させることのできるディスプレイ群も維持する。さらに、オペレータインタフェース200は、アナウンスメントオペレーションすなわちメッセージと「OK」ボタンだけの単純なポップアップを表示する一方的なメッセージを提供する。

10

20

【0030】

データハンドラ210は、他のAPCシステム100のコンポーネントが集めたデータを受け取り、これらのデータをデータベースサーバ115、117上のデータストア230、232(リレーションナルデータベースなど)に格納する。データハンドラ210は、標準的な構造化照会言語(SQL)コマンドを受け取ることのできるものであってもよいし、あるいは、データハンドラ210は、異なるタイプのアクセスプロトコルを翻訳してSQLコマンドまたは他の何らかのプロトコルコマンドを生成するものであってもよい。データ格納機能を中心を集めることで、さまざまなコンポーネントの移植性が高くなる。

【0031】

次に、図2に示す簡略ブロック図を参照してベースライン制御スクリプト152、154の大まかなオペレーションについて説明する。同図において、ベースライン制御スクリプト152、154と多様な共有ベースラインライブラリとの間のリンクが示されている。総じて言うと、ベースライン制御スクリプト152、154は、APCシステム100内の制御スクリプトを開発するためのフレームワークを提供するものである。ベースライン制御スクリプトはライブラリに格納された共有ベースラインコンポーネントを利用する。図示の実施形態では、共有ベースラインコンポーネントは、制御アルゴリズムを定義するための制御ベースラインライブラリ240と、共通に使われる数学関数(sum、mean、medianなど)を定義するための数学ベースラインライブラリ250と、スクリプト実行の通信アスペクト(マシンインタフェース195、オペレータインタフェース200、他のそのような外部コンポーネントを介してのデータストア230、232、マシンインタフェース195、機器インタフェース160との対話)を定義するための対話ベースラインライブラリ260と、共有の共通関数を定義するためのユーティリティベースラインライブラリ270と、施設に特有の他のライブラリ240、250、260、27ライインライブラリ280と、ベースのルーチンに対する例外または関数を定義するための施設ライブラリ280と、ベースライン制御スクリプトに対する呼出に含まれる特定のオペレーションIDからのレイヤー(例、ポリゲート層)を定義するためのレイヤーライブラリ290とを含む。ベースラインライブラリ240、250、260、270、280、290を、ベースライン制御スクリプト152、154のオペレーションの間にコントロールエグゼキュータ220によってリンクさせておいてもよい。

30

40

【0032】

総じて言うと、ベースライン制御スクリプト152、154は、スクリプトに対する呼出

50

に含まれる情報と施設ベースラインライブラリ 280 およびレイヤーベースラインライブラリ 290 に含まれる情報に基づいてコントロールアクションの性質を判定する。ベースライン制御スクリプト 152、154 は、制御ベースラインライブラリ 240 とリンクして必要な制御関数にアクセスする。ベースライン制御スクリプト 152、154 は対話ベースラインライブラリ 260 とリンクし、コントロールアクションを実施したりツール 120、125 の動作レシピの更新目的で機器インタフェース 160 と通信したりするのに使われるデータを集めるための関数にアクセスする。数学ベースラインライブラリ 250 の関数を必要に応じてベースライン制御スクリプト 152、154 または他のライブラリの他の関数から呼ぶようにしてもよい。

10

【0033】

ここで図 3 を参照すると、ベースラインプロセススクリプト 152 の構成を示す簡略プロック図が示されている。ベースラインプロセススクリプト 152 は、アプリケーションコンフィギュレーションブロック 300 と、ベースラインアプリケーションセットアップブロック 310 と、コントローラ定数およびコンテキスト固有の設定ブロック 320 と、フィードフォワードデータ分析ブロック 330 と、制御スレッドブロック 340 と、危機 (jeopardy) ブロック 350 と、コントロールアクションおよびビジネスルールブロック 360 と、結果ブロック 370 とを含む。

【0034】

アプリケーションコンフィギュレーションブロック 300 内には、機器インタフェース 160 からの呼出に含まれる情報に基づいてコントローラが使うユーザグローバルコンフィギュレーション変数が定義されている。これは、レシピ管理システム (RMS) からの変数の値 (すなわちレシピ設定のグローバルデータベース) と必要なコンテキスト変数とを含む。コンテキスト変数の値は制御スレッドを定義し、一般にツール識別コード、ロット番号、オペレーション番号などの変数の値で構成されている。また、必要なベースライン変数も一定の値である。一例として、エラー通知用の電子メール一覧、タイムアウトの値、「子」ロットとみなされる 1 ロットで許容される最大ウエハ数、コントローラが使った前のレイヤ情報 (フィードフォワード情報) などがあげられる。

20

【0035】

ベースラインアプリケーションセットアップブロック 310 は、アプリケーションコンフィギュレーションブロック 300 でのセット時にロット番号とウエハ数量の値を利用し、ロット番号、ファミリ名、親名、施設、ウエハ数、ステータス (すなわちロットが親ロットであるか子ロットであるか) の値を返す。また、ベースラインアプリケーションセットアップブロック 310 は、コントローラからのポップアップウインドウとすべてのポップアップウインドウタイトルの最初の部分の送り先である端末のリストを設定する。

30

【0036】

コントローラ定数およびコンテキスト固有の設定ブロック 320 はすでに定義されたコンテキストと RMS 情報とを利用して、制御の動きを計算するのにコントローラが使う値を設定する。たとえば、コントローラ定数およびコンテキスト固有の設定ブロック 320 は、RMS に定義された値に基づいて、コンテキスト情報 (または「スレッド」指定) を使って制御モデルパラメータの値を設定することができる。その具体的な一例は、特定のエッチャンバのコンテキストに基づいて制御モデルで使われるエッチャレートの値や、RMS に定義されているようなエッチャンバのエッチャレートの値を設定することであろう。また、コントローラ定数およびコンテキスト固有の設定ブロック 320 は、アプリケーションコンフィギュレーションブロック 300 に設定されているようなロット番号およびレイヤ名ごとのクエリを使ってデータベースからフィードフォワード情報を読み出す。

40

【0037】

フィードフォワードデータ分析ブロック 330 は、特定のロットに関連したデータのアレイに含まれるエレメントをチェックし、抜けている値にはデフォルト値を埋める。たとえば、前のプロセスのターゲットを使って、コントローラが使うフィードフォワード情報の一部として必要な抜けている測定値を設定することができる。デフォルト値を使わずに抜

50

けているフィードフォワード情報の値を設定するための他の方法をフィードフォワードデータ分析ブロック 330 で実施するようにしてもよい。

【0038】

制御スレッドブロック 340 は、データストア 230、232 を照会して現在の制御スレッドに関連した制御状態を読み出すのに必要なキーと状態構造の値を設定する。これらのキーはデータストア 230、232 からスレッド状態データを読み出すのに使われる。制御スレッドブロック 340 は、このスレッドコンテキストで処理された新しいロットの順に並べられたデータのスタックの中のスレッド状態データを検索する。このような値が見つかった場合、これは制御モデルを含むユーザ定義関数に渡され、この関数でスレッド状態の値を計算して返す。スタックに値が見つからなかった場合、制御スレッドブロック 340 は階層を上がって検索し、スレッド状態の値のある最初の階層レベルからデータを読み出す。スタックおよびすべての階層レベルは同様ではあるが精度の異なるデータを含むと仮定される。10

【0039】

危機ブロック 350 は、データベース内でルックアップを実行し、危機スタックのロット数（すなわち、最後の計測オペレーション以降に特定のスレッドで処理されたロットのスタック）についての値を読み出す。この値はこの危機カテゴリ内のロット数についての閾値すなわち一般に RMS で指定される値と比較される。この閾値以下である場合、コントローラはそのまま残る。閾値を超えた場合、危機スタックのロットのリストからのひとつのロットについて計測イベントを実施するようオペレータに指示するポップアップディスプレイを表示してコントローラは異常終了する。20

【0040】

コントロールアクションおよびビジネスルールブロック 360 はコントローラの核心である。コントロールアクションおよびビジネスルールブロック 360 は状態およびターゲット情報からコントローラ入力（プロセスレシピ更新）を計算する。これらの結果はグローバル制御結果アレイに置かれる。次に、コントロールアクションおよびビジネスルールブロック 360 はビジネスルールを実行し、コントローラのユーザ入力オーバーライドに従ってプロセスレシピ更新のチェックを制限および／またはプロセスレシピ更新を設定する。30

【0041】

結果ブロック 370 は、プロセスレシピ更新、データ計算／フォーマットを含むコントロールアクションおよびビジネスルールブロック 360 からの出力あるいはイベントを受け、これをまとめて一時的に蓄積し、データをフォーマットする。結果ブロック 370 は一時に蓄積したデータを機器インターフェース 160 に送信し、機器インターフェース 160 に対してマシンインターフェース 195 によってセットアップ／スタートマシン呼出を開始する。次に、結果ブロック 370 は、現在のコンテキスト（スレッド）についてロット番号およびレイヤに対してデータストア 230、232 に格納されたデータを格納する。危機スタックも最後の計測イベント以降に処理された追加のロットとして現在のロットで更新される。40

【0042】

図4を参照すると、ベースライン計測スクリプト 154 の構成を示す簡略ブロック図が示されている。ベースライン計測スクリプト 154 は、計測ツールセットアップブロック 400 と、アプリケーションコンフィギュレーションブロック 410 と、ベースラインアプリケーションセットアップブロック 420 と、受入れツールデータブロック 430 と、コントローラ定数およびコンテキスト固有の設定ブロック 440 と、制御スレッドブロック 450 と、モデル更新ブロック 460 と、結果ブロック 470 とを含む。

【0043】

計測ツールセットアップブロック 400 内では、データ収集を開始して一時的に蓄積されていたデータをコントロールエグゼキュータ 220 に送信する。機器インターフェース 165 へのマシン呼出をセットアップ／開始するためにマシンインターフェース 190 も開始さ50

れる。

【0044】

アプリケーションコンフィギュレーションブロック410およびベースラインアプリケーションセットアップブロック420は、ベースラインプロセススクリプト152に関連して上述した同じ名前のブロックと同様の機能を果たす。

【0045】

受入れツールデータブロック430は、ベースライン計測スクリプト154を一時停止し、一般に計測ツールであるデータソースからのデータセットを待つ。このイベントの待ち時間とスクリプトの一時停止を解除するイベントの名前が受入れツールデータブロック430に指定されている。
10

【0046】

コントローラ定数およびコンテキスト固有の設定ブロック440は、ベースラインプロセススクリプト152に関連して上述した同じ名前のブロックと同様の機能も果たす。

【0047】

制御スレッドブロック450は、現在のスレッドについて算出した制御状態をデータストア230、232に格納するのに必要なキーおよび状態構造の値を設定する。また、制御スレッドブロック450は、スレッド状態を更新するのに必要なすべての値を計算する。この関数は、定義されたグローバル変数を読み取り、必要とされる結果を計算する。これらの結果には、ロット平均、プロセスレート、ターゲットまたは予測からの偏差など、コントローラの更新時に使用される統計値または値が含まれる。この関数の結果はグローバル制御結果アレイにおかれる。
20

【0048】

モデル更新ブロック460は、ビジネスルール、スペックリミットチェックング、コントローラのオーバーライドを実施するのに使われる。この関数は、定義されたグローバル変数を読み取り、最終結果を設定する。モデル更新ブロック460は、コントローラを更新するのに使用される値ならびに制御履歴に記録される値の設定を担っている。この関数の結果はグローバル制御結果アレイにおかれる。

【0049】

結果ブロック470は、コントローラ定数およびコンテキスト固有の設定ブロック440からの出力を受け、これを一時的に蓄積し、機器インタフェース165と互換性を持つようデータをフォーマットする。ベースライン計測スクリプト154によるデータ出力も制御履歴ファイルに書き込まれる。供給された変数名に基づいて制御履歴のヘッダが生成される。ログファイルにはファイルの1行目でエンコードされたヘッダがある。計算されたヘッダがファイルの1行目と整合しない場合、既存のファイルの名前を変え、新しいものを開始する。
30

【0050】

ここで図5を参照すると、本発明のもうひとつの実施形態による、单一処理ツール120上に複数のコントロールアクションを実装できる複数コントローラ用ベースラインプロセススクリプト500の簡略ブロック図が示されている。たとえば、フォトリソグラフィ用ステッパーにオーバーレイコントローラと隔壁寸法コントローラの両方を持たせることができる。これらのコントローラは、処理済みウエハからのフィードバックを利用して、露光線量、露光時間、焦点などのさまざまなステッパバラメータを調節する。ポリシリコン層を形成するためのツールなどの成膜ツールにもポリシリコンの粒度やポリシリコン層の厚さなどのバラメータを制御するためのコントローラを複数持たせることができる。
40

【0051】

ベースラインプロセススクリプト500が呼ばれると、このスクリプトは呼出しに含まれる情報に基づいて必要なコントロールアクションを判定する。ロットが処理されるコンテキストは、動作対象のコントローラがどれになるかを決める。このコンテキストは、オペレーションID、エンティティID、製品ID、個々の実行（ラン）についての要件を決める他の別個の識別子によって定義される。まず、エンティティIDを使ってツールのタイ
50

の大雑把なクラスを求める（ステッパ、エッチャー、炉など）。たとえば、エンティティ ID が処理ツール 120 をステッパであると識別した場合、ステッパ制御コードが呼出される。

【0052】

ステッパ制御コードの中で、スクリプト内のコンテキスト変数がチェックされ、どの個別コントローラが呼ばれるかを判定する。オペレーション ID は、プロセスが走ることを示している（ポリゲートマスク vs 第2の層間誘電体層マスク（ILD）など）。各コントローラはコンテキスト状況のセットに適用され、これらのコンテキスト条件がすべて満たされた場合にのみ走る。たとえば、CD コントローラは、ポリゲートマスク用に走ることができるが、第2の ILD マスクプロセスでは走ることはできない。一方、オーバーレイコントローラは、両方のマスクイベントで走ることができる。10

【0053】

ベースラインプロセススクリプト 500 は、ツールセット（ステッパなど）に基づいて必要なツールコードを整合させるための柔軟性を提供し、利用できるコントローラ（オーバーレイ、CD など）をすべて走らせる準備をする。同じメインスクリプトを走らせ、同じサブルーチンを呼ぶことが可能であるが、現在のコンテキストで必要なコントローラのみが起動される。

【0054】

複数コントローラ用ベースラインプロセススクリプト 500 は、アプリケーションコンフィギュレーションブロック 510 と、ベースラインアプリケーションセットアップブロック 520 と、コントローラ定数およびコンテキスト固有の設定ブロック 530 と、フィードフォワードデータ分析ブロック 540 と、制御スレッドブロック 550 と、危機ブロック 560 と、コントロールアクションおよびビジネスルールブロック 570 と、結果ブロック 580 とを含む。複数コントローラ用ベースラインプロセススクリプト 500 は、後述する点を除いてベースラインプロセススクリプト 152 と同じように動作する。20

【0055】

コントローラ定数およびコンテキスト固有の設定ブロック 530 は、どのコントローラを適用可能であるか（コントローラ A、コントローラ B あるいは両方など）を判定し、過去に定義されたコンテキストおよび RMS 情報を利用して、制御の動きを算出するのに各コントローラが使う値を設定する。また、コントローラ定数およびコンテキスト固有の設定ブロック 530 は、アプリケーションコンフィギュレーションブロック 510 に設定されているようなロット番号およびレイヤ名ごとのクエリを使って必要なコントローラ各々のフィードフォワード情報をデータベースから読み出す。フィードフォワードデータ分析ブロック 540 は、特定のロットに関連したデータのアレイ中のエレメントをチェックし、各コントローラについて抜けている値のデフォルト値を埋める。30

【0056】

制御スレッドブロック 550 は、データストア 230、232 を照会してアクティブなコントローラ各々の現在の制御スレッドに関連した制御状態を読み出すのに必要なキーと状態構造の値を設定する。これらのキーはデータストア 230、232 からスレッド状態データを読み出すのに使われる。制御スレッドブロック 550 は、このスレッドコンテキストで処理された新しいロットの順に並べられたデータのスタックのスレッド状態データを検索する。このような値が見つかった場合、これは制御モデルを含むユーザ定義関数に渡され、この関数でスレッド状態の値を計算して返す。スタックに値が見つからなかった場合、制御スレッドブロック 340 は階層を上がって検索し、スレッド状態の値のある最初の階層レベルからデータを読み出す。スタックおよびすべての階層レベルは同様ではあるが精度の異なるデータを含むと仮定される。40

【0057】

コントロールアクションおよびビジネスルールブロック 570 は、各コントローラの状態およびターゲット情報からコントローラ入力（プロセスレシビ更新）を計算する。使用されるコントローラは複数あるため、ひとつのコントローラがそのコントロールアクション50

を判定する上で頼りにしている状態情報に他のコントローラが影響を与える場合がある。したがって、それぞれのコントロールアクションを判定する順序を決めるための相対的な優先値をコントローラに割り当てるようにしてもよい。2番目のコントローラの状態情報データを、優先度の高いコントローラがそのコントロールアクション決定に基づいて更新することができる。次に修正された状態情報に基づいて2番目のコントローラがそのコントロールアクションを判定する。このようにして連携することで、コントローラは動作レシピの変更に関して互いに競合することがなくなる。

【0058】

結果ブロック580は、アクティブなコントローラすべてからコントロールアクション出力を集め、このデータを一時的に蓄積し、データをフォーマットする。結果ブロック580は一時に蓄積したデータを機器インターフェース160に送信し、機器インターフェース160に対してマシンインターフェース195によってセットアップ/スタートマシン呼出を開始する。次に、結果ブロック580は、現在のコンテキスト（スレッド）についてロット番号およびレイヤに対してデータストア230、232に格納されたデータを格納し、危機スタックを更新する。

10

【0059】

ここで図6を参照すると、本発明の他の実施形態による複数のコントローラを統合するための方法の簡略流れ図が示されている。ブロック600では、複数のツールで被加工物を処理する。ブロック610では、（制御実行マネージャ150によるなどの方法で）複数のツールのうちの選択したツールについてベースライン制御スクリプトを開始する。
20 ベースライン制御スクリプトの開始後、コントロールエグゼキュータ220が残りのタスクを実施する。ブロック620では選択したツールに必要な制御ルーチン群を特定する。ブロック630では、必要な制御ルーチン群について選択したツールに関連した過去のコントロールアクションに関する制御状態情報を読み出す。ブロック640では、必要な制御ルーチン群からの最初の制御ルーチンを実行し、最初のコントロールアクションを生成する。ブロック650では、最初のコントロールアクションに基づいて必要な制御ルーチン群からの2番目の制御ルーチンに関連した制御状態情報を変更する。ブロック660では、修正後の制御状態情報に基づいて2番目の制御ルーチンを実行し、2番目のコントロールアクションを生成する。

20

【0060】

本発明は本願明細書の教示内容の利益を享受する当業者らに自明の上記とは異なるが等価な方法で改変および実施することのできるものであるため、上記にて開示した個々の実施形態は一例にすぎない。さらに、添付の請求の範囲に記載したものを除き、本願明細書に図示した構造または設計の詳細に対する限定を何ら意図するものではない。したがって、上記にて開示した個々の実施形態を変更または改変してもよく、そのような変形例はいずれも本発明の範囲に包含されるとみなせることは明白である。よって、本願明細書で求める保護は添付の請求の範囲に記載のとおりである。

30

【図面の簡単な説明】

【0061】

【図1】本発明の一実施形態による高度プロセス制御（A P C）システムの簡略ブロック図である。

40

【図2】図1のシステム内のベースライン制御スクリプトと、多様な共有ベースラインライブラリとの間のリンクを示す図である。

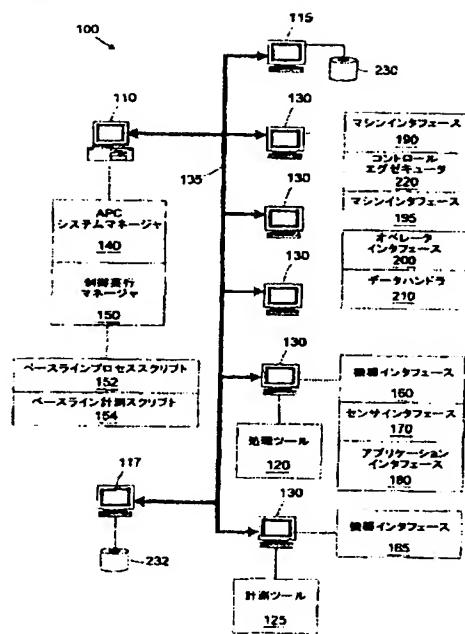
【図3】ベースラインプロセススクリプトの構成を示す簡略ブロック図である。

【図4】ベースライン計測スクリプトの構成を示す簡略ブロック図である。

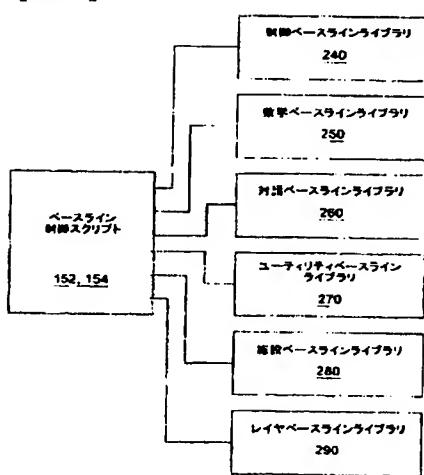
【図5】複数コントローラ用ベースラインプロセススクリプトの構成を示す簡略ブロック図である。

【図6】本発明のもうひとつの実施形態による複数のコントローラを統合するための方法の簡略流れ図である。

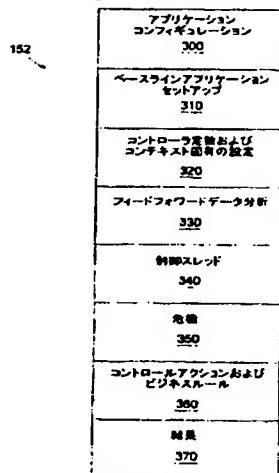
【図 1】



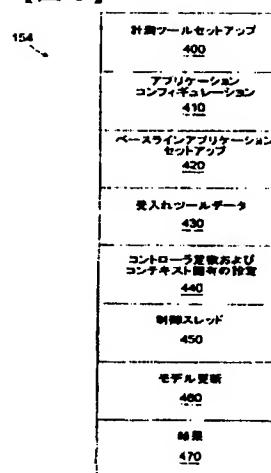
【図 2】



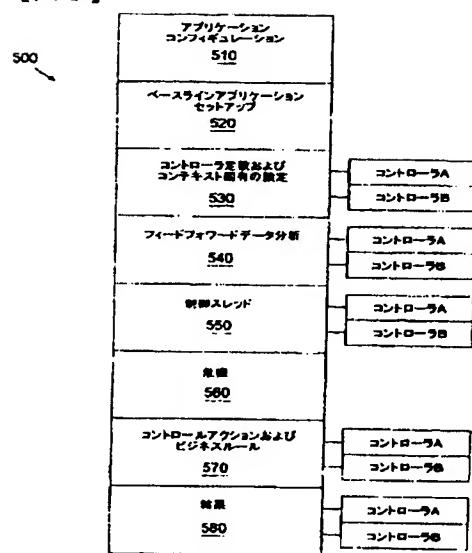
【図 3】



【図 4】



【図 5】



【図 6】

